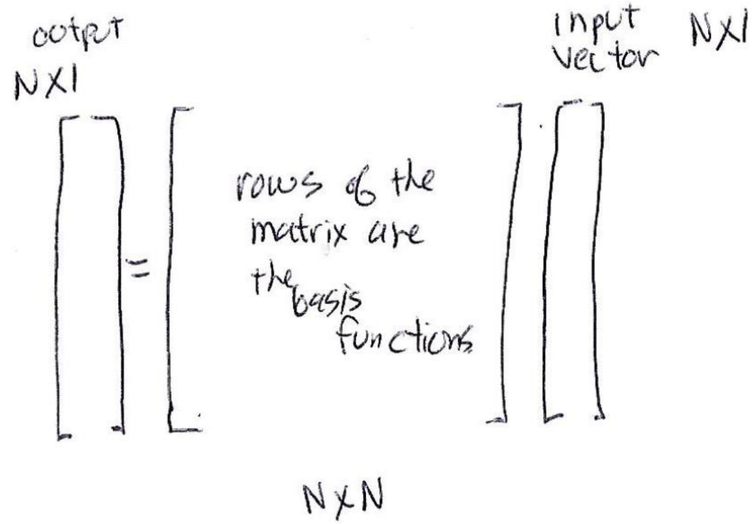
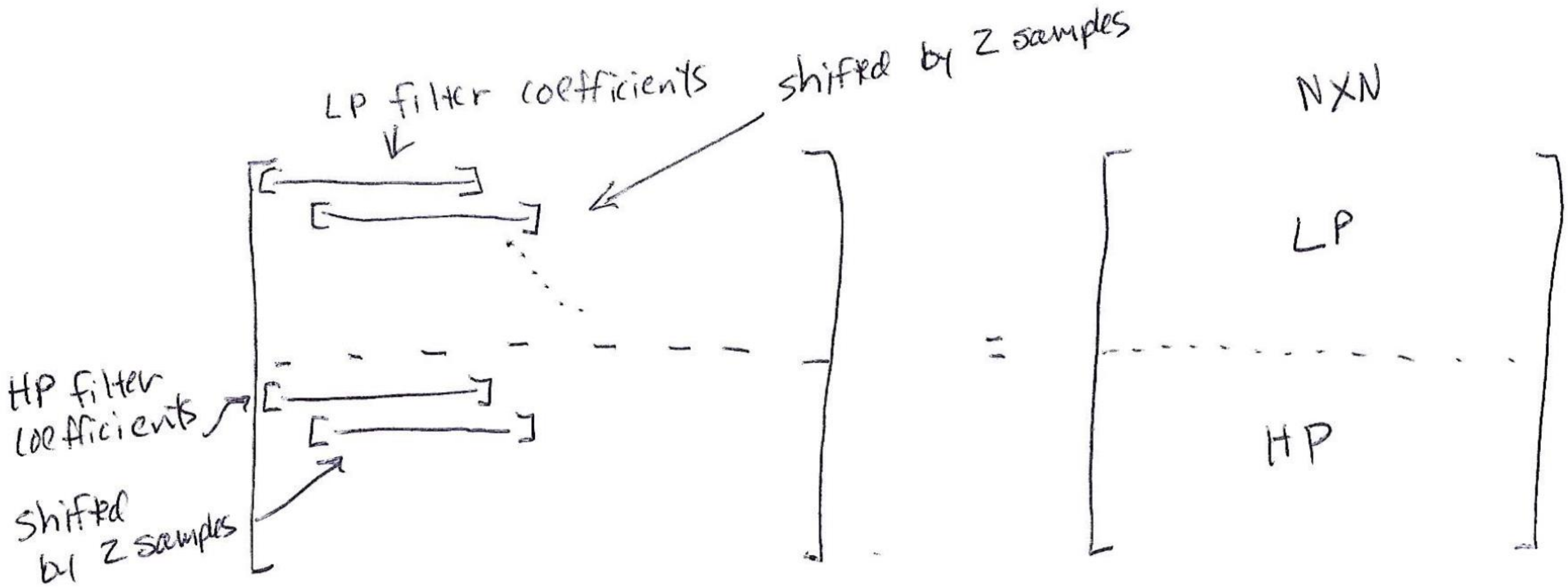
For any transform



matrix for
one level
of the WT



Forward wavelet Transform matrices

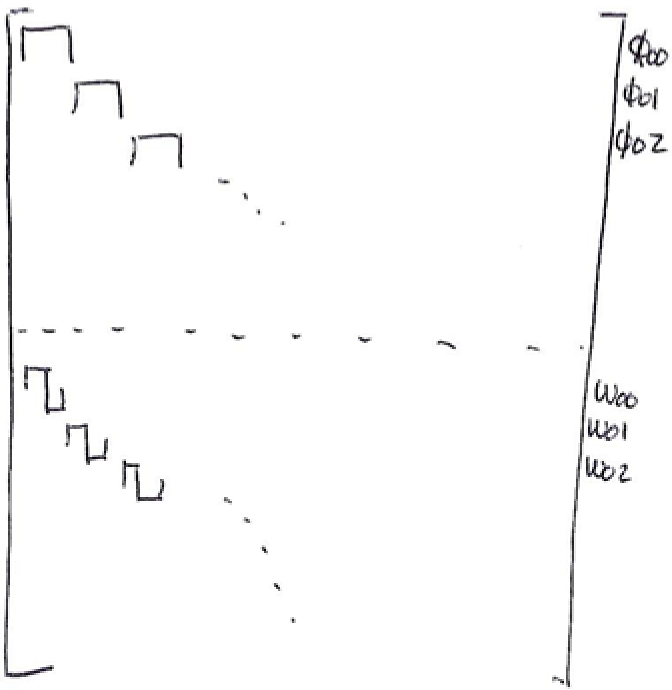
matrix for two levels of the transform

$$
\begin{array}{cc} N \times N & N \times N \\
\left[\begin{array}{c:c} LP & 0 \\ HP & \\ \hdashline 0 & I \end{array}\right] &
\left[\begin{array}{c} LP \\ \hdashline HP \end{array}\right]
\end{array}
=
\begin{array}{c} N \times N \\
\left[\begin{array}{cc} \phi_0 & \text{scaling function at scale 0} \\ \hdashline W_0 & \text{wavelets at scale 0} \\ \hdashline W_1 & \text{wavelets at scale 1} \end{array}\right]
\end{array}
$$

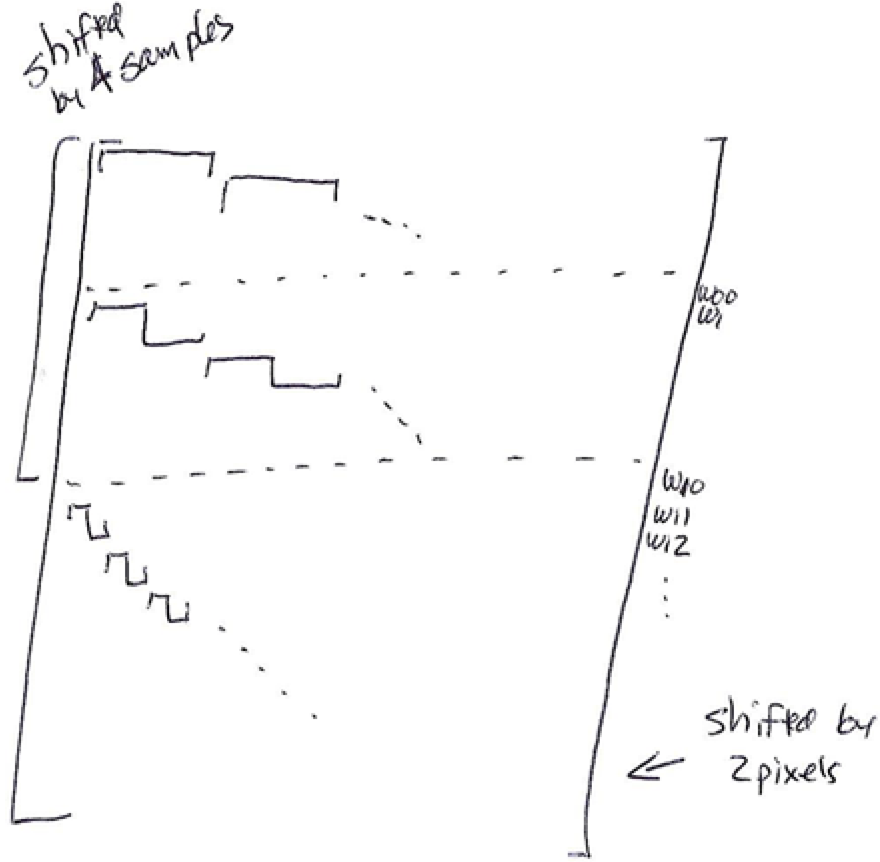
matrix for three levels

$$
\left[\begin{array}{c:c:c} LP & 0 & 0 \\ HP & & \\ \hdashline 0 & I & \\ \hdashline 0 & & I \end{array}\right]
\left[\begin{array}{c:c} LP & 0 \\ HP & \\ \hdashline 0 & I \end{array}\right]
\left[\begin{array}{c} LP \\ \hdashline HP \end{array}\right]
=
\left[\begin{array}{cc} \phi_0 & \text{at scale 0} \\ \hdashline W_0 & \text{at scale 0} \\ \hdashline W_1 & \text{at scale 1} \\ \hdashline W_2 & \text{wavelets at scale 2} \end{array}\right]
$$

Haar forward transform matrices

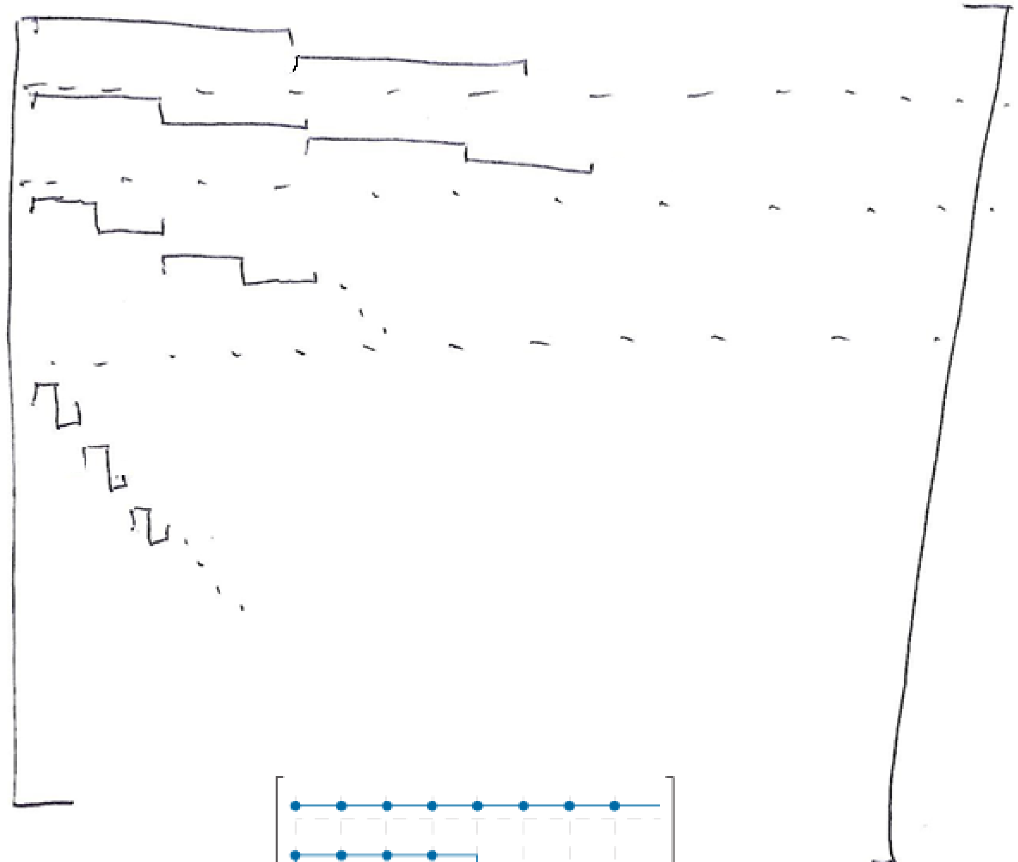


Haar matrix for 1 level



Haar matrix for 2 levels

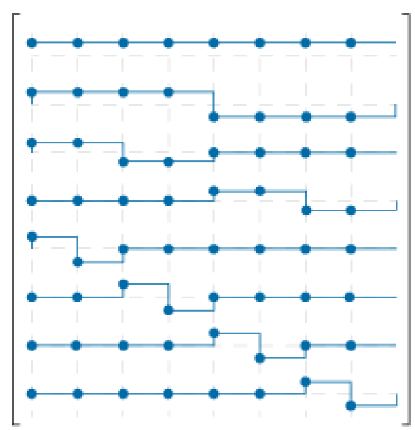
3 levels



shifted by 8 samples

shifted by 4 samples

shifted by 2 samples



Two-dimensional Transform From One-dimensional Transform

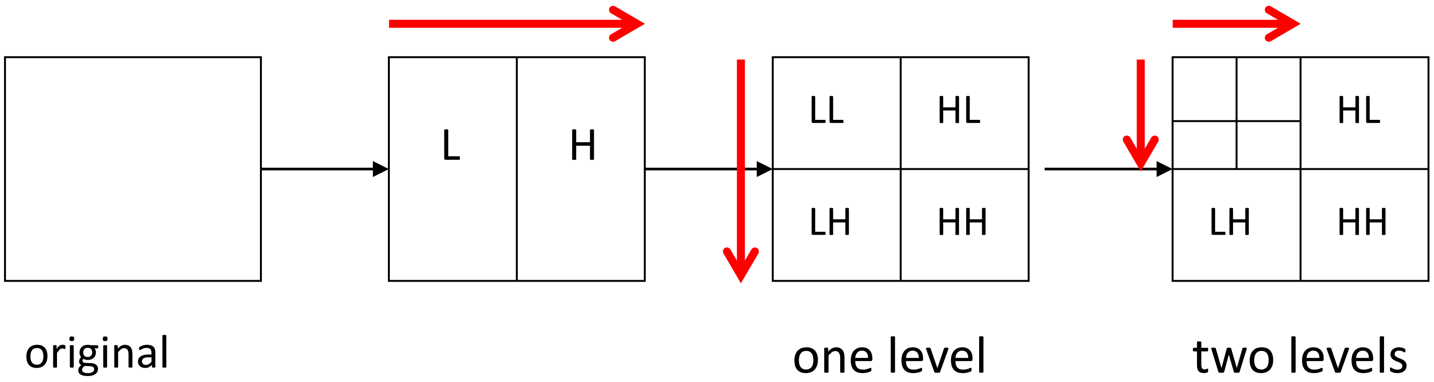
A 2-D WT can be calculated as follows:

Step 1: Perform the 1-D WT on each row.

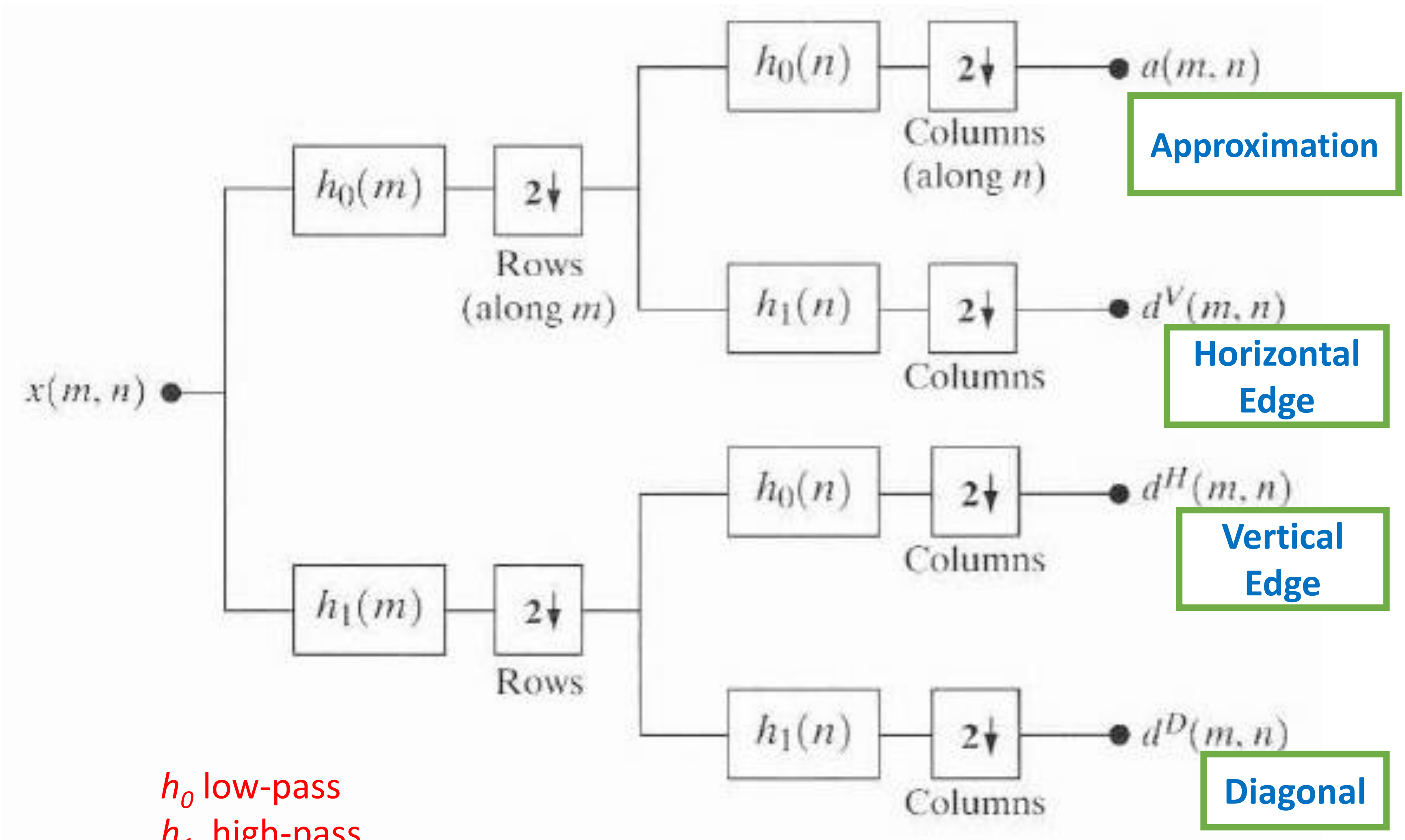
Step 2: Perform the 1-D on each column (completes one level)

Step 3: Repeat steps (1) and (2) on the lowest subband (LL) for the next level

Step 4 repeat for as many levels as desired

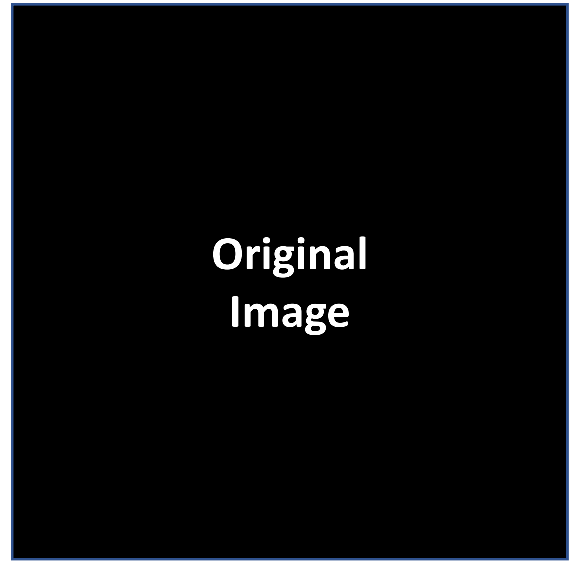


Two-Dimensional Algorithm
Extension of One-Dimension



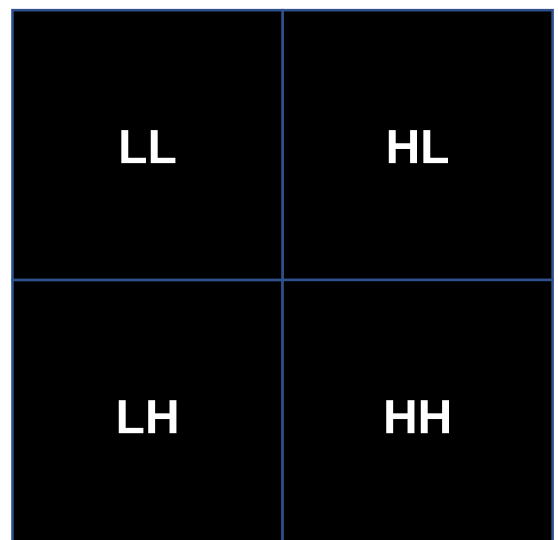
h_0 low-pass
h_1 high-pass

2-D Transform (1 level)



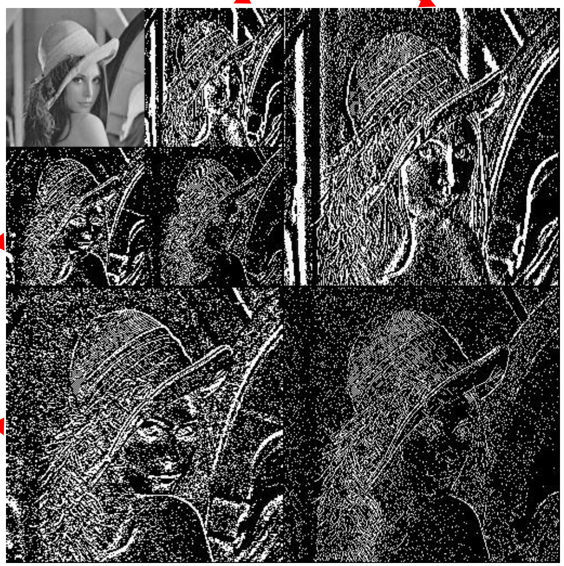
Vertical emphasis

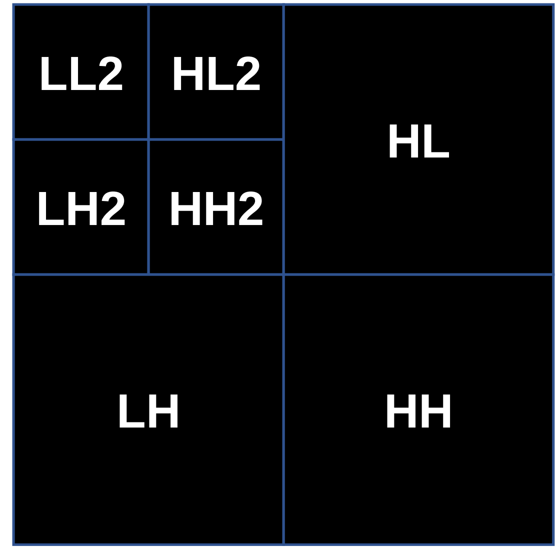
Horizontal emphasis



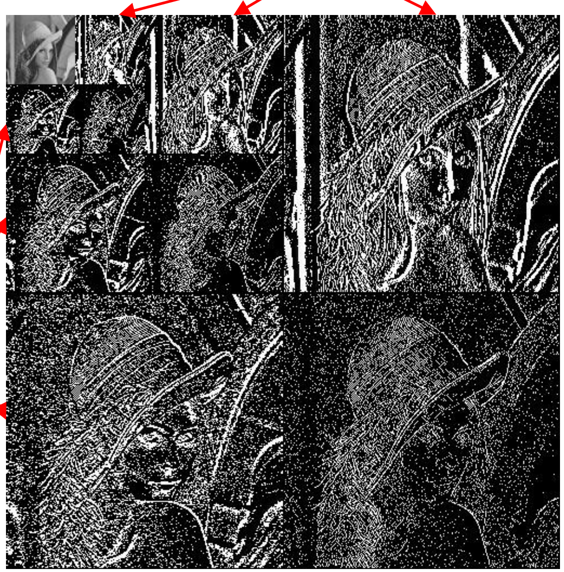
2-D Transform (2 and 3 levels)

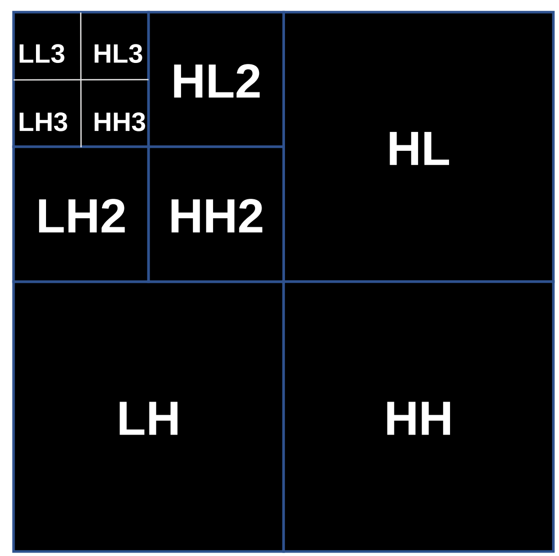
Emphasis of different scales





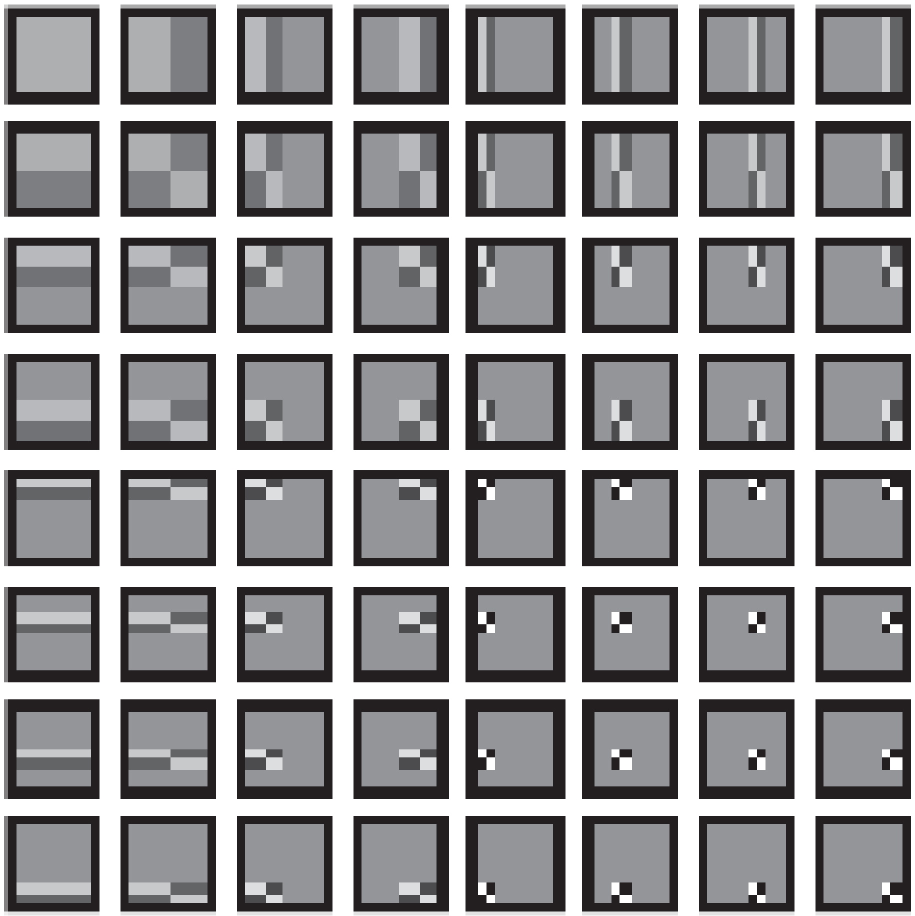
Emphasis of different scales





Haar 2-D basis functions for 8 x 8 image (3 levels)

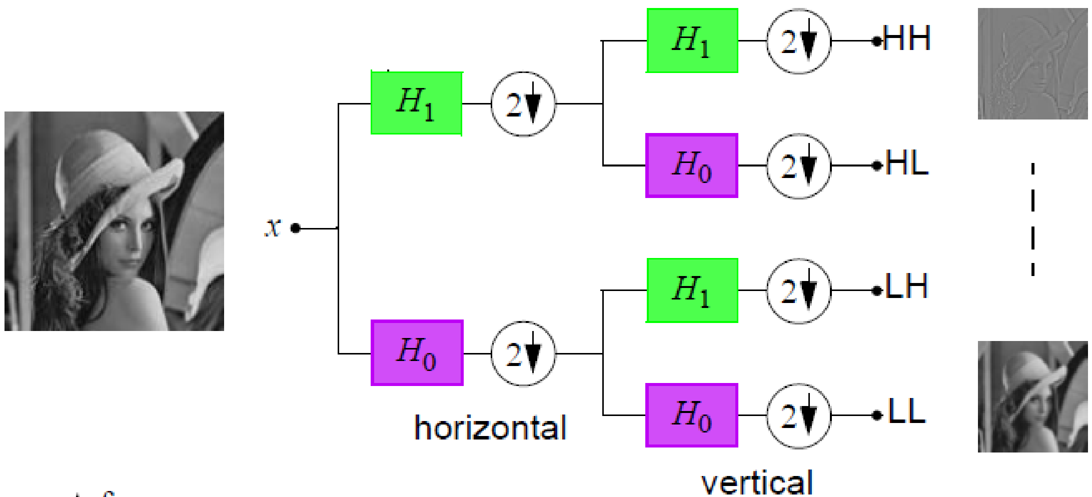
Basis images for 8 x 8 image - each basis image is 8 x 8

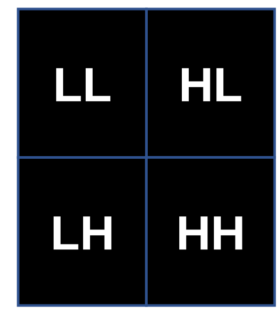


Multiply an input image by each basis image (element-by-element) to get a wavelet coefficient

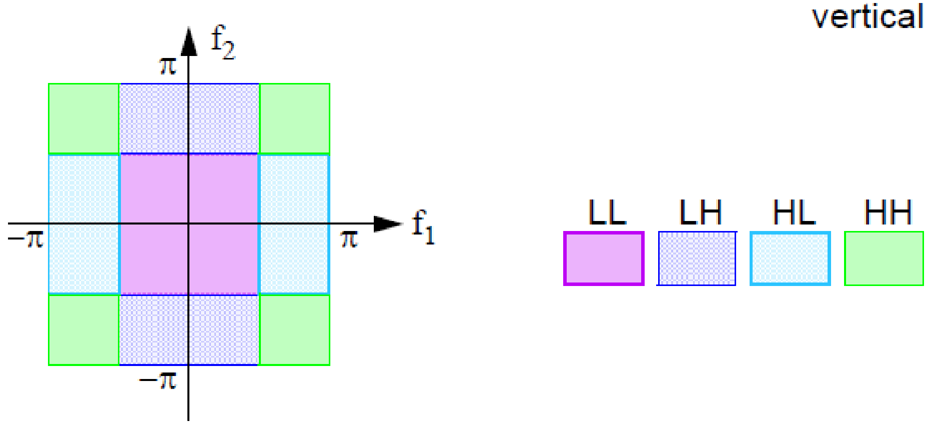
2-D Frequency Representation

One level of wavelet transform





Spatial domain



Fourier ranges of subband images

H_0 low-pass
H_1 high-pass

Edge Detection Using Wavelet Transform

a b
c d
e f

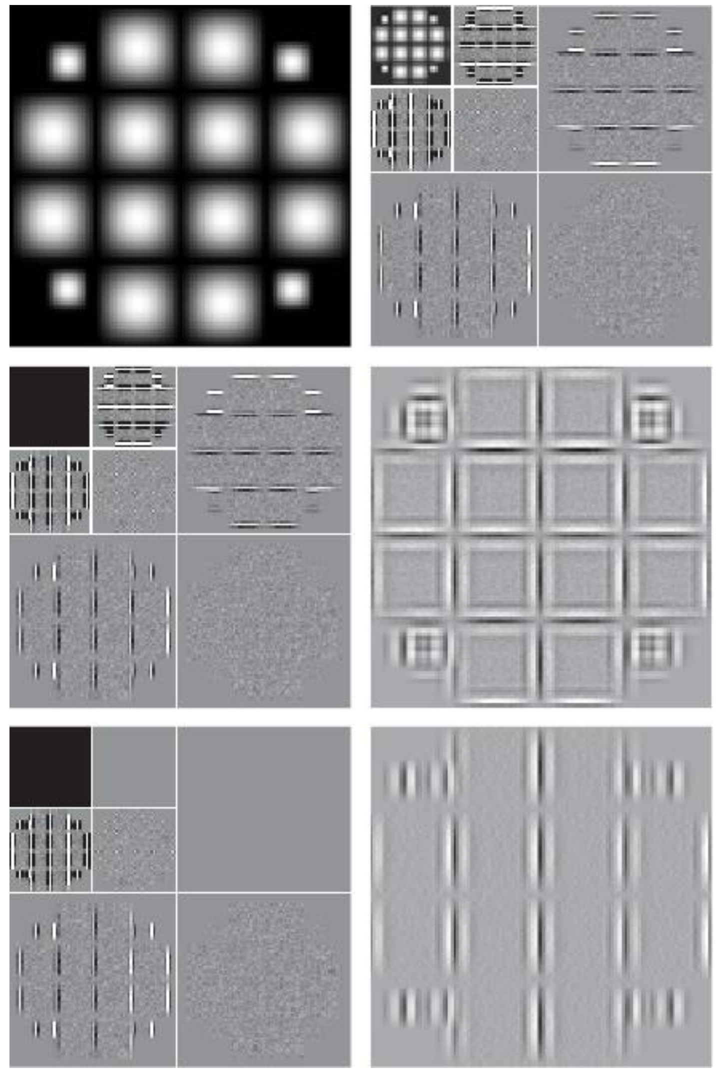


FIGURE 6.32 Modifying a DWT for edge detection: (a) orginal image; (b) two-scale DWT with respect to 4th-order symlets; (c) modified DWT with the approximation set to zero; (d) the inverse DWT of (c); (e) modified DWT with the approximation and horizontal details set to zero; and (f) the inverse DWT of (e). (Note when the detail coefficients are zero, they are displayed as middle gray; when the approximation coefficients are zeroed, they display as black.)

Image Compression is Efficient

- Why image compression?

- A 3504 x 2336 pixel (full color) image contains,

  3504 x 2336 x 3 pixels = 24,556,032 Bytes

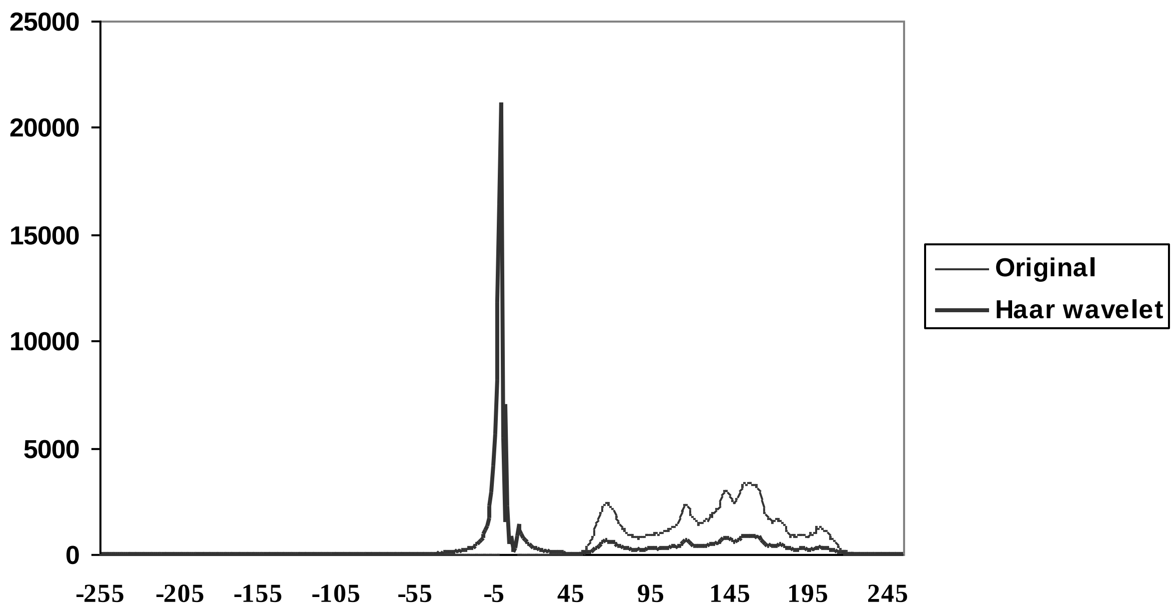
  = 23.4 Mbyte

- Goal

  Reduce the redundancy of the image data in order to store or transmit data in an efficient form.

Wavelets Compact Energy Well

- Wavelet coefficient histogram



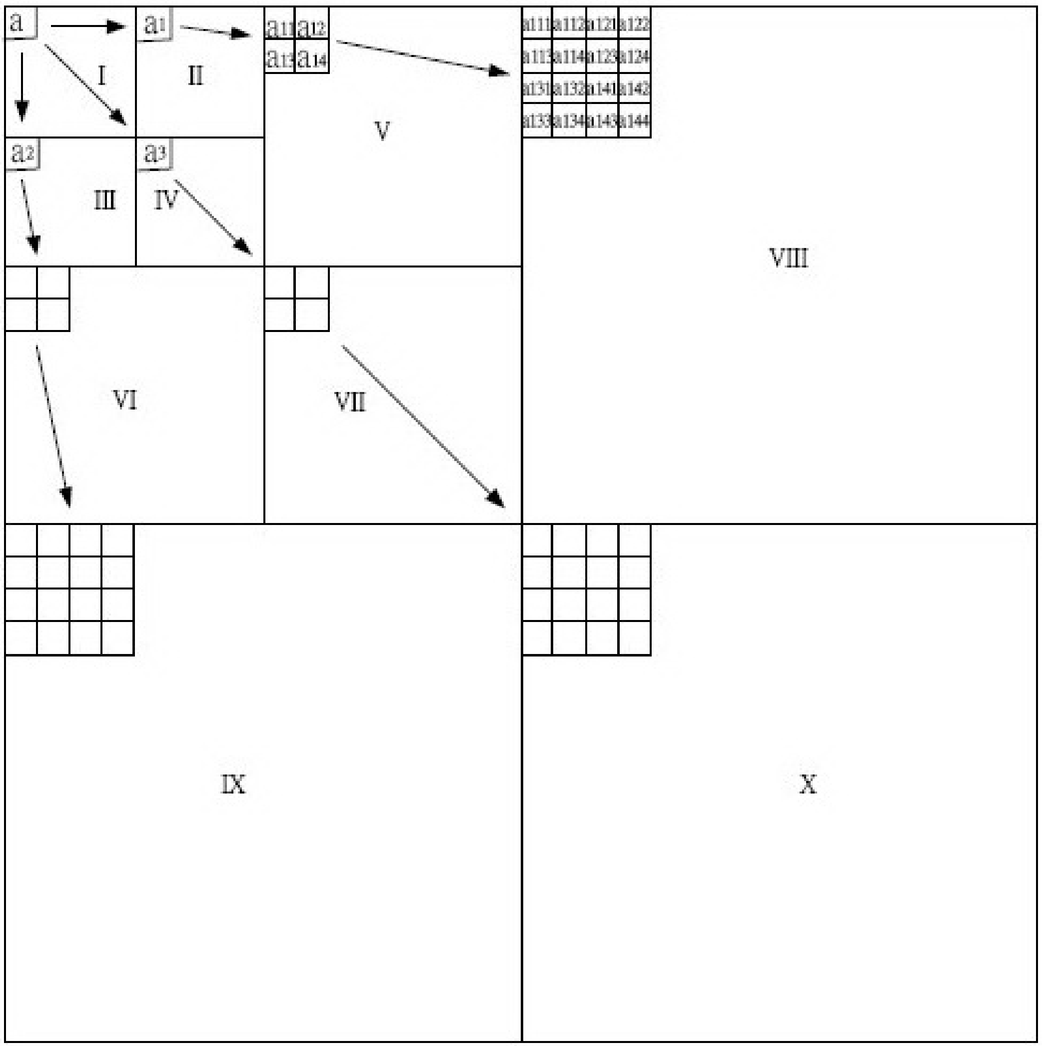
- Wavelet Transform of image can be represented with smaller coefficients than original
- Can remove smallest coefficients with little loss of quality – image compression

Smoother Wavelets Often Give Better Results

- Coefficient entropies (measurement of uncertainty)

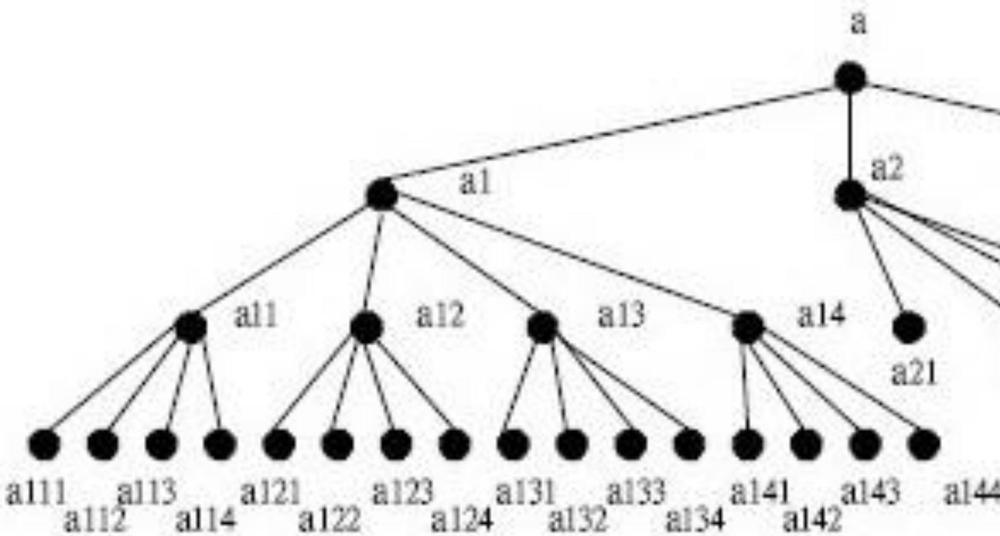
	Entropy
Original image	7.22
1-level Haar wavelet	5.96
1-level linear spline wavelet	5.53
2-level Haar wavelet	5.02
2-level linear spline wavelet	4.57

Wavelets Allow Zerotree Coding That Further Improves Compression

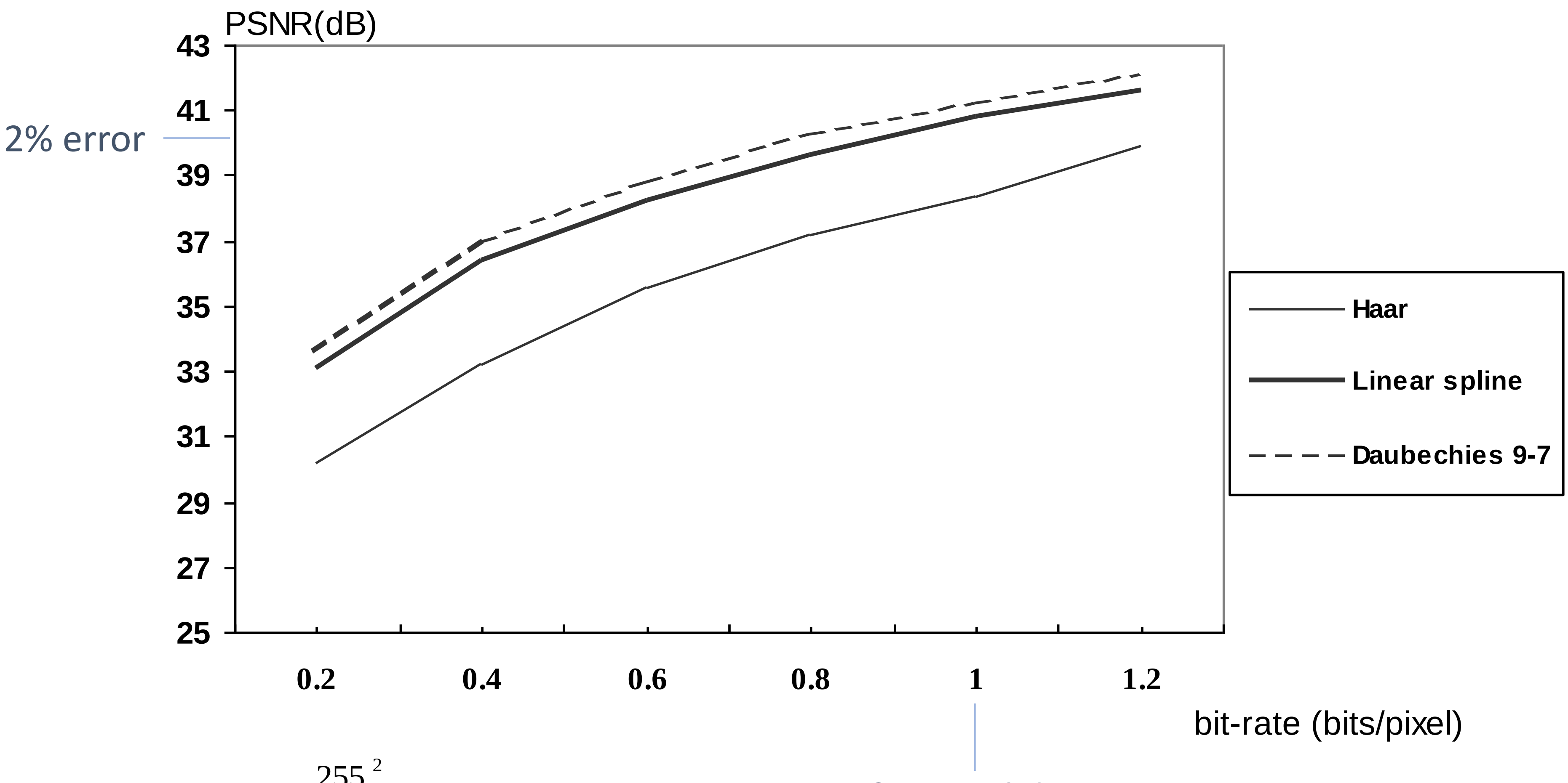


Zerotree coding groups pixels that are similar

Grouping Similar Pixels Improves Compression



Substantial Compression Results



$$PSNR\,(dB) = 10\log_{10}\frac{255^2}{mse}$$

Typical Results

Original

SPIHT 0.2 bits/pixel

JPEG 0.2 bits/pixel

Denoising Usually Applied to Additive Noise

- Finite Length Signal with Additive Noise:

$$y[n] = x[n] + \sigma\, r[n],\ 0 \le n \le N-1$$

Observations of x + noise

Signal

Standard Deviation

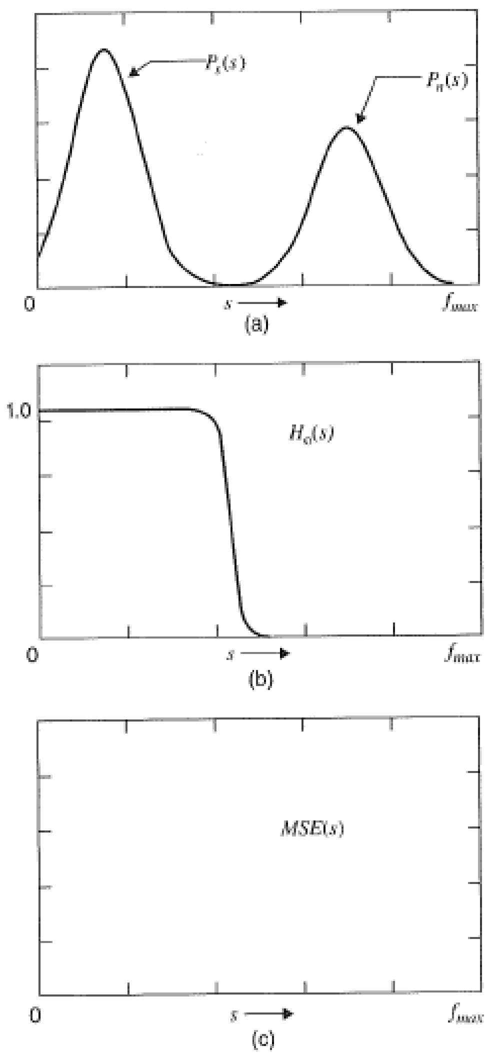
Noise : $r[n] \sim N(0,1)$

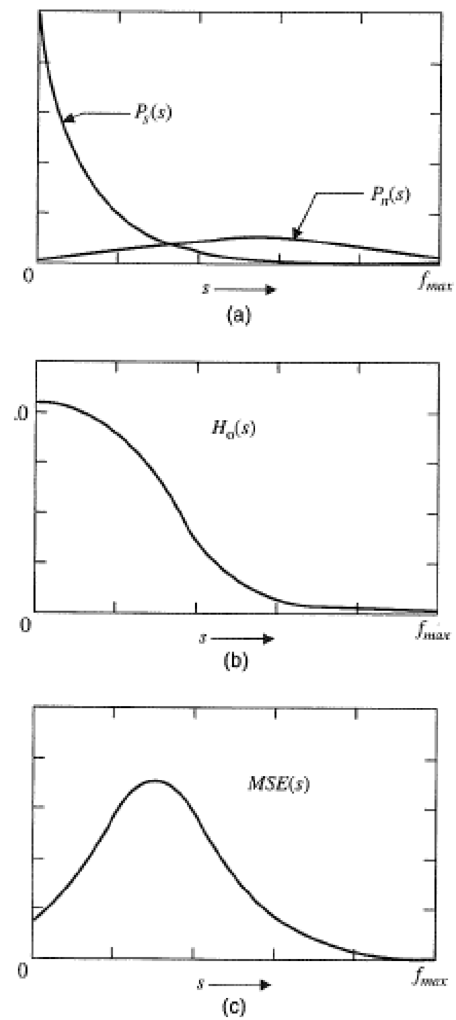
The Wiener Filter is the Optimal Linear Filter for Removing Noise

Optimal linear filter in terms of MSE if statistics of signal and noise are known

Wiener filter for uncorrelated signal and noise:
P_S – power spectrum of signal
P_N – power spectrum of noise.

$$H_o(s) = \frac{P_s(s)}{P_s(s) + P_n(s)} \qquad s \neq 0$$





Power spectra

Wiener filter

MSE between filtered signal and noise-free signal

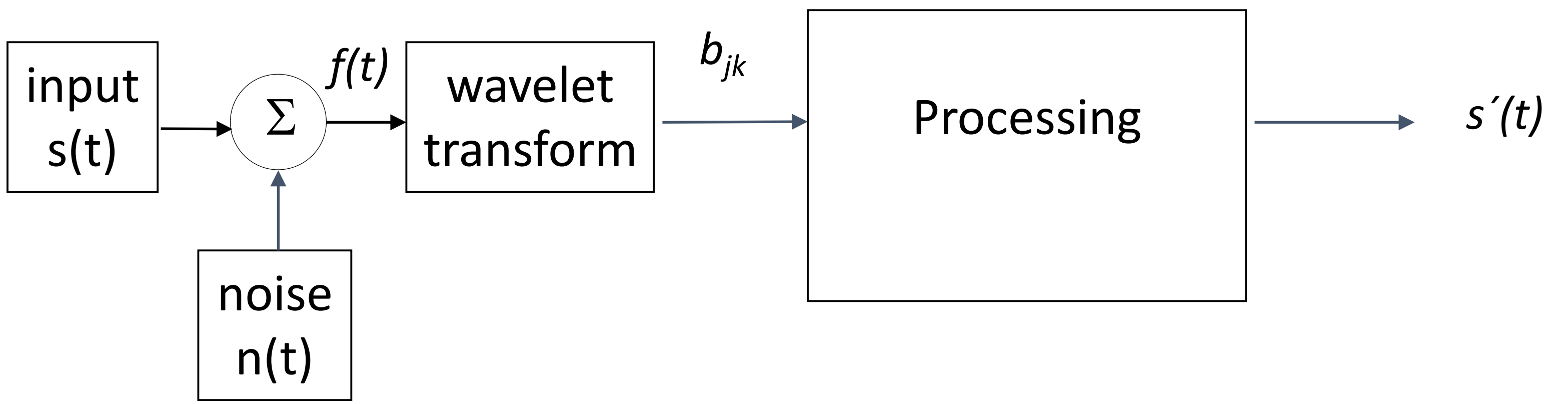
The Wiener Filter Works Well but May Blur Edges

original

noisy

Wiener filtered
(optimal linear filter)

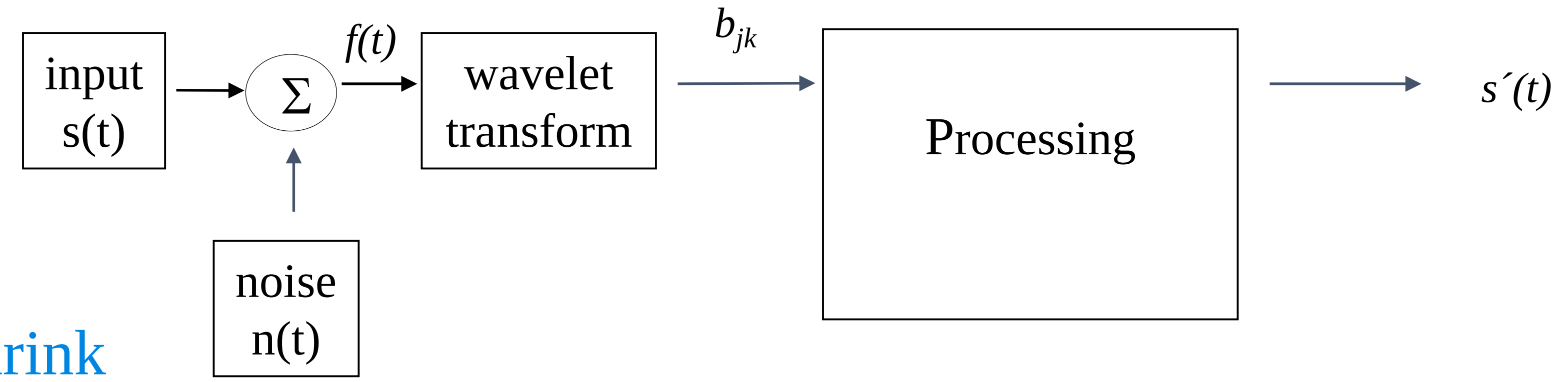
Denoising is Similar to Image Compression



Many different methods

Some notable ones:
- VisuShrink
- BayesShrink
- BLS-GSM
- BM3D

VisuShrink – Simple and Works Well



VisuShrink

- Based on assumption of additive Gaussian white noise that has $\mu = 1$ and $\sigma = 1$.

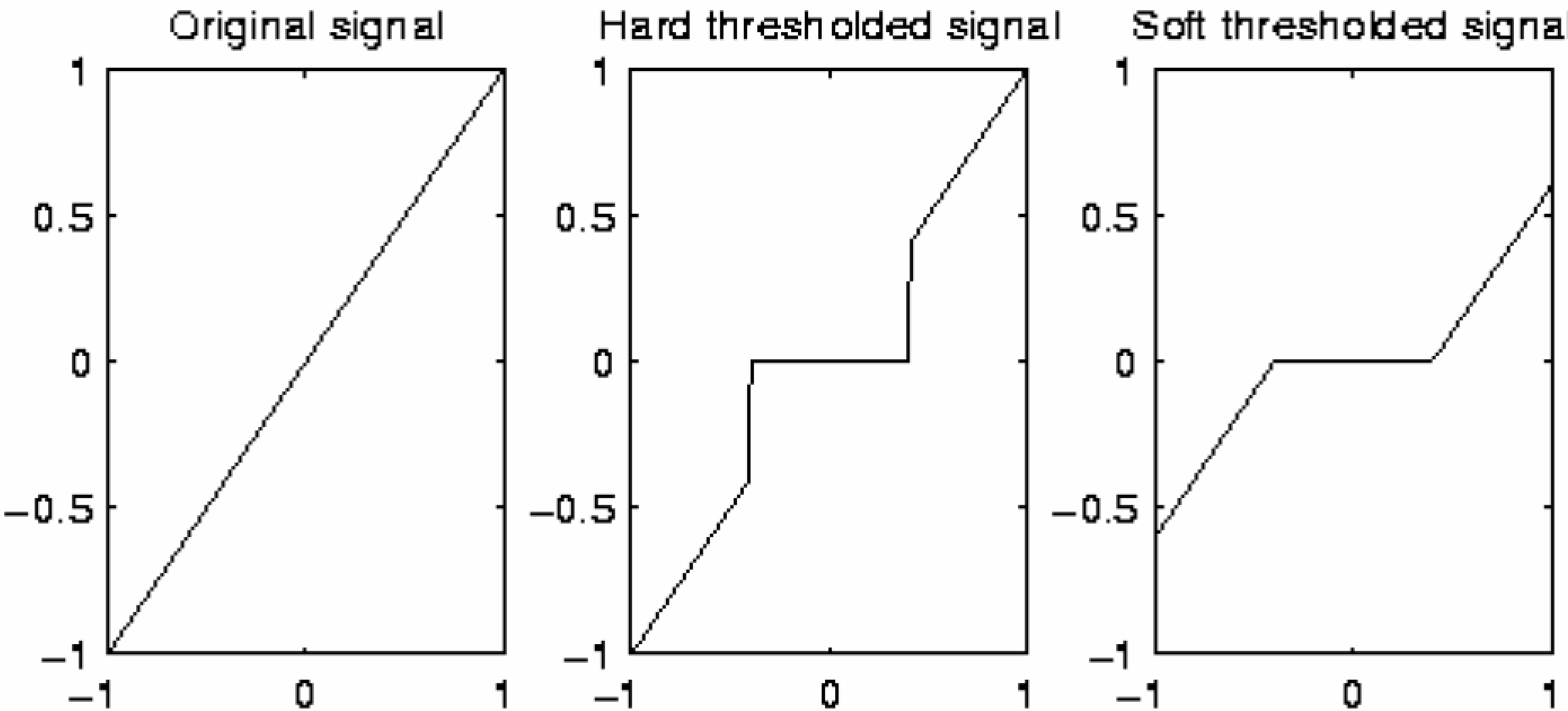
- White noise under any orthogonal transform is still white noise (can subtract noise using a soft-threshold).

- Threshold $T = (\sigma/.6745)\sqrt{(2 \log (N))}$ where σ can be estimated from the noisy signal.

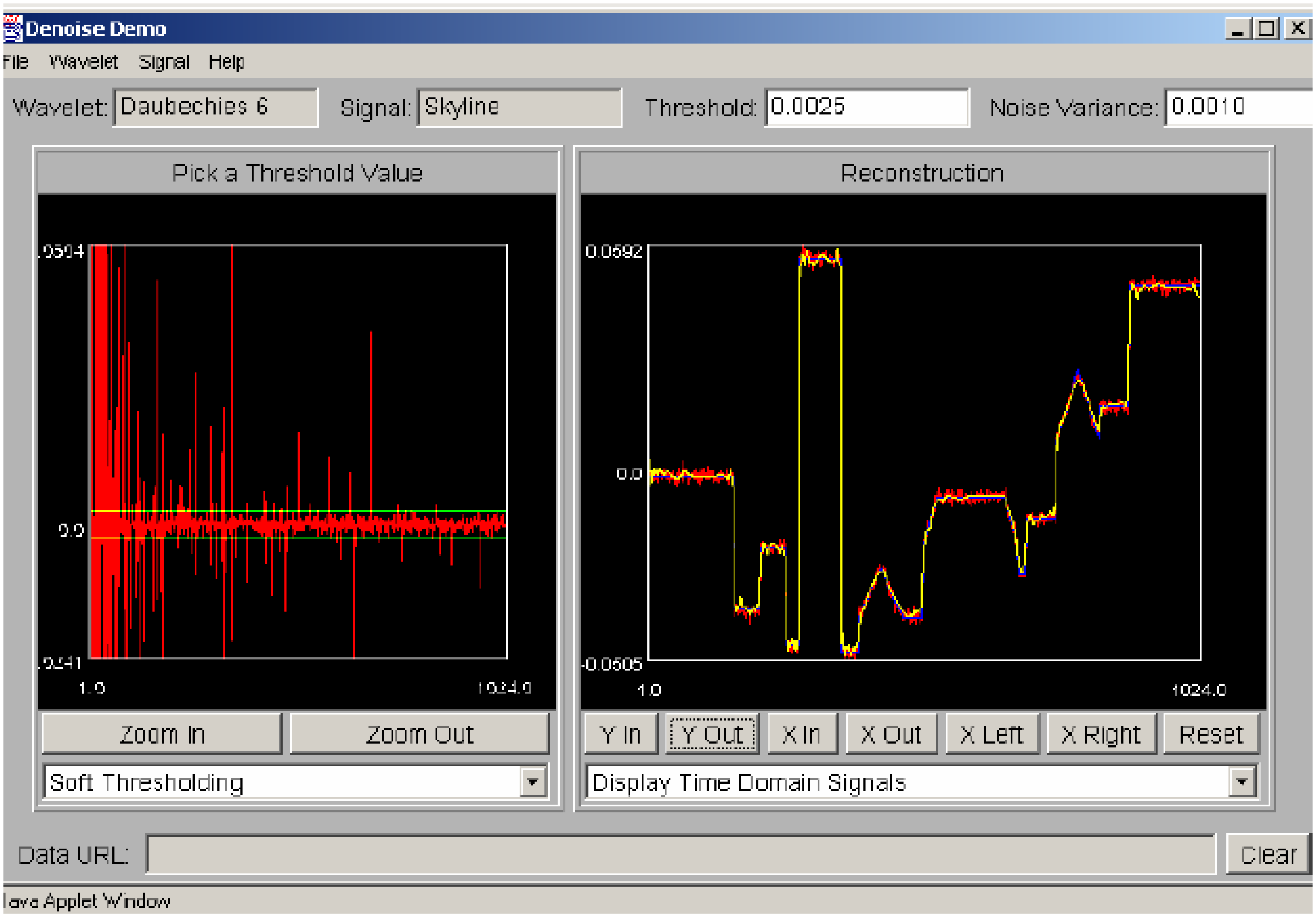
- SureShrink usually performs better. It uses a hybrid between VisuShrink and Stein's unbiased risk estimator (SURE).

D. L. Donoho, I. M. Johnstone, "Ideal spatial adaptation via wavelet shrinkage," *Biometrika*, vol. 81, 425-455 (1994)

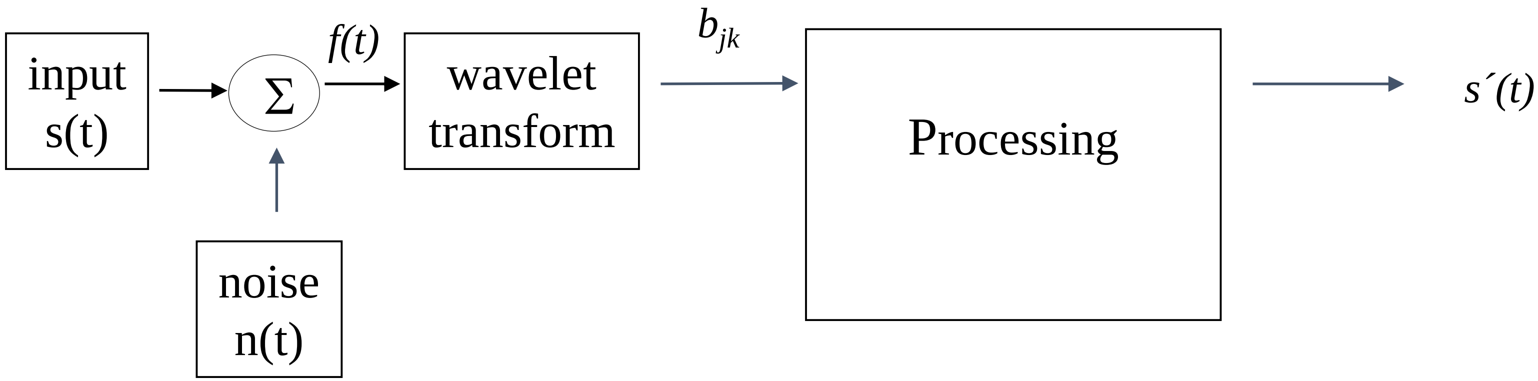
Soft-Thresholding Often Used



Visushrink in Matlab



Denoising – BayesShrink is Based on Noise in Each Subband



BayesShrink

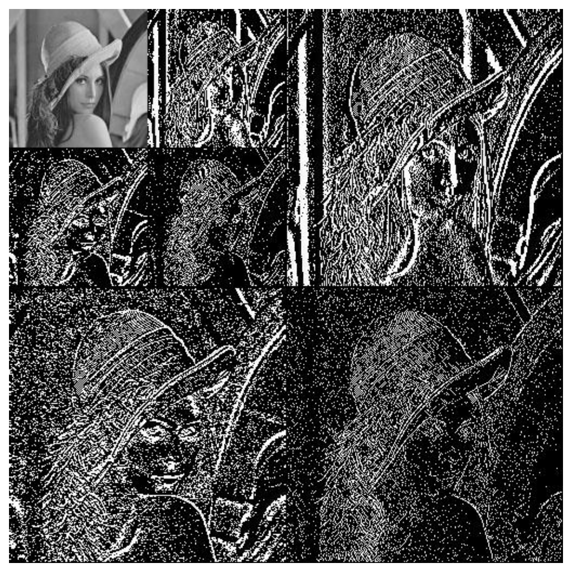
•Relies on data-driven parameters in Bayesian framework.

•Assumes wavelet coefficients in a subband can be summarized by a general Gaussian distribution.

•Threshold $T_i = \sigma^2/\sigma_{xi}$ for the ith subband where σ is the noise estimate at the finest scale, and σ_{xi} is signal standard deviation estimate.

S. G. Chang, B. Yu, M. Vetterli, "Adaptive wavelet thresholding for image denoising," *IEEE Trans. on Image Proc*, vol. 9(9) (2000)

BayesShrink -Different Threshold for Each Subband



σ_1

σ_2

σ_3

Better Denoising – BM3D

BM3D – Block matching 3D

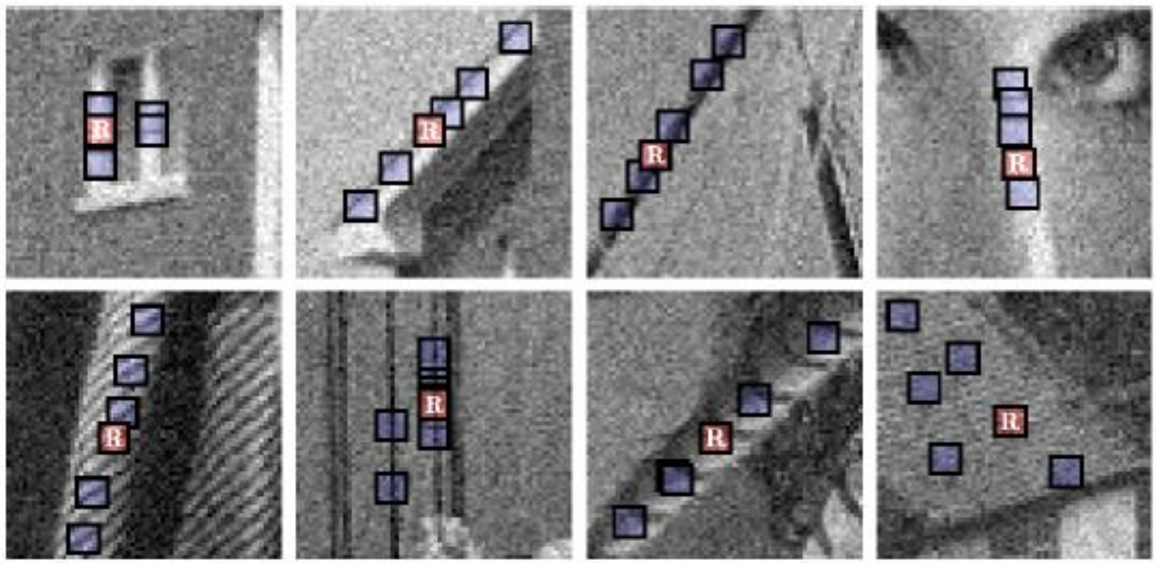


Figure 14: Illustration of grouping blocks from noisy natural images corrupted by white Gaussian noise with standard deviation 15 and zero mean. Each fragment shows a reference block marked with "R" and a few of the blocks matched to it.

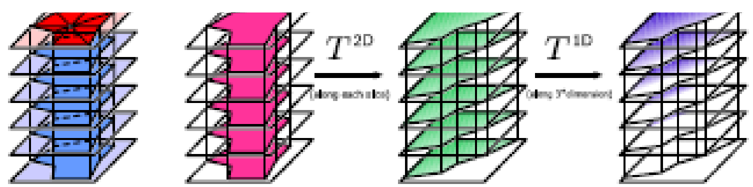


Figure 16: Shape-adaptive grouping and the forward shape-adaptive transform used in the collaborative filtering of the group.

K. A. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian,."Image denoising by sparse 3D transform-domain collaborative filtering," *IEEE Trans. on Imag. Proc.,* 16(8), 2080 – 2095 (2007).

Good Results Using BM3D Approach

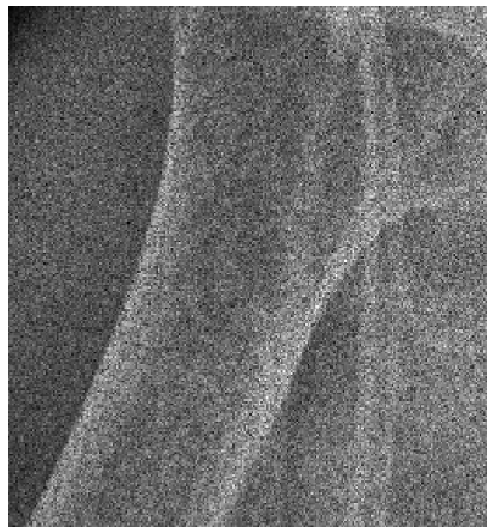
Conventional
x-ray image

X-ray image
with reduced
dose

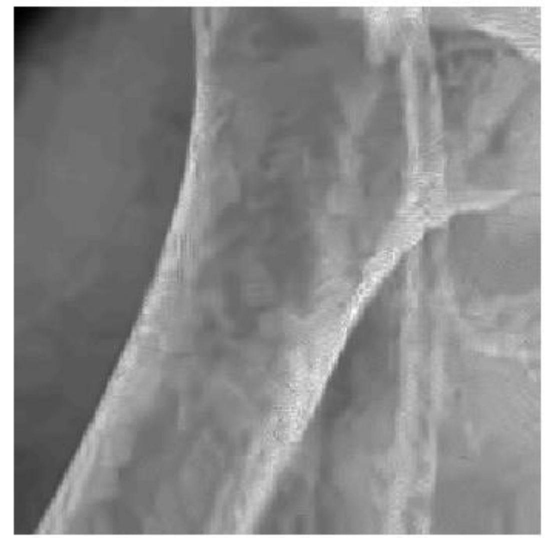
Dennoised
with
BM3D

Good Results Using BM3D Approach

Conventional
x-ray image



X-ray image
with reduced
dose



Dennoised
with
BM3D

Matlab commands

In the wavelet toolbox

Forward transform

[C,L] = wavedec(x, N, ‘wname’)

x is the input signal

N is the number of decomposition levels

‘wname’ is the name of the wavelet from the command *wfilters.* Alternatively, it can be replaced directly with low-pass and high-pass filter coefficients with two vectors as: LP,HP

C is the wavelet decomposition - it may be longer than x because the algorithm is most likely using convolution rather than a matrix with wrap-around

L is a vector containing the number of coefficients at each level – it contains N + 2 numbers

L(1) = # of scaling function coefficients at what we called level 0 in class. They are in positions 1 through L(1) in C.

L(2) = # of wavelet coefficients at what we called level 0 in class. They are in the next L(2) positions: L(1) + 1 through L(1) + L(2) in C.

L(3) = # of wavelet coefficients at what we called level 1 in class. They are in the next L(3) positions.

L(i) = # of wavelet coefficients at what we called level i-2 in class

L(N+2) = N

Inverse transform

x = waverec(C, L, ‘wname’)

The definitions of C, L, and ‘wname’, are as in the wavedec command and should match it for proper reconstruction.